CS 540 Introduction to Artificial Intelligence
Linear Models & Linear Regression

Yingyu Liang
University of Wisconsin-Madison
Oct 12, 2021
Based on slides by Fred Sala
Outline

• Unsupervised Learning: Density Estimation
  – Kernel density estimation: high-level intro

• Supervised Learning & Linear Models
  – Parameterized model, model classes, linear models, train vs. test

• Linear Regression
  – Least squares, normal equations, residuals, logistic regression
Short Intro to Density Estimation

Goal: given samples $x_1, \ldots, x_n$ from some distribution $P$, estimate $P$.

- Compute statistics (mean, variance)
- Generate samples from $P$
- Run inference
Simplest Idea: Histograms

Goal: given samples $x_1, \ldots, x_n$ from some distribution $P$, estimate $P$.

Define bins; count # of samples in each bin, normalize
Simplest Idea: Histograms

Goal: given samples $x_1, \ldots, x_n$ from some distribution $P$, estimate $P$.

**Downsides:**

i) High-dimensions: most bins empty

ii) Not continuous

iii) How to choose bins?
Kernel Density Estimation

Goal: given samples $x_1, \ldots, x_n$ from some distribution $P$, estimate $P$.

**Idea:** represent density as combination of “kernels”

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K \left( \frac{x - x_i}{h} \right)$$

- Center at each point
- Kernel function: often Gaussian
- Width parameter
Kernel Density Estimation

**Idea:** represent density as combination of kernels

- “Smooth” out the histogram
Break & Quiz

Q 1.1: Which of the following is not true?

- A. Using a Gaussian kernel for KDE, all possible values for $x_i$ will have non-zero probability.
- B. The goal of KDE is to approximate the true probability distribution function of $X$.
- C. When using a histogram, every bucket must be represented explicitly in memory
- D. With some kernels, KDE can assign zero probability to some subset of values for $x_i$. 

Break & Quiz

**Q 1.1:** Which of the following is not true?

- A. Using a Gaussian kernel for KDE, all possible values for $x_i$ will have non-zero probability.
- B. The goal of KDE is to approximate the true probability distribution function of $X$.
- C. When using a histogram, every bucket must be represented explicitly in memory
- D. With some kernels, KDE can assign zero probability to some subset of values for $x_i$. 
Break & Quiz

**Q 1.1:** Which of the following is not true?

- A. Using a Gaussian kernel for KDE, all possible values for $x_i$ will have non-zero probability. *(Gaussian PDF positive for all inputs)*
- B. The goal of KDE is to approximate the true probability distribution function of $X$. *(same goal as histograms)*
- C. **When using a histogram, every bucket must be represented explicitly in memory**
- D. With some kernels, KDE can assign zero probability to some subset of values for $x_i$. *(Consider $K = \text{uniform}(0,1)$)*

About C: we don’t necessarily need to keep every bucket explicitly. For example, we can keep only those that are non-empty.
Back to Supervised Learning

Supervised learning:
- Make predictions, classify data, perform regression
- Dataset: \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)
  - Features / Covariates / Input
  - Labels / Outputs
- Goal: find function \(f : X \rightarrow Y\) to predict label on new data
Back to Supervised Learning

How do we know a function $f$ is good?

- Intuitively: “matches” the dataset $f(x_i) \approx y_i$
- More concrete: pick a loss function to measure this: $\ell(f(x), y)$
- Training loss/empirical loss/empirical risk
  \[
  \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)
  \]
  
  - Find a $f$ that minimizes the loss on the training data (ERM)

Three key elements:
1. A class of functions from which we choose our model
2. A loss function measuring the match between the prediction and the true label
3. A method to minimize the training loss
Loss Functions

What should the loss look like?

- If \( f(x_i) \approx y_i \), should be small (0 if equal!)
- For classification: 0/1 loss \( \ell(f(x), y) = 1\{f(x_i) \neq y_i\} \)
- For regression, square loss \( \ell(f(x), y) = (f(x_i) - y_i)^2 \)

Others too! We’ll see more.
If we consider choosing our model among the family of all (measurable) functions, then we can have the constructed function which has zero training loss but can have large error on future test inputs.

The phenomenon that a function gets much lower training loss than on test inputs is called overfitting. It usually happens when we choose model from a too large family of functions. Intuitively, it’s likely that a very large family of functions contains some function that can fit the training data but not the test data.
So we should choose a more specific class of functions. This lecture considers the family of linear functions.

\[ f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \theta_0 + x^T \theta \]
Training The Model

- Parametrize it by weights/parameters
- Minimize the loss

\[
\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)
\]

Best parameters = best function \( f \)

Linear model class \( f \)

Square loss

\[
= \frac{1}{n} \sum_{i=1}^{n} \ell(\theta_0 + x_i^T \theta, y_i)
\]

\[
= \frac{1}{n} \sum_{i=1}^{n} (\theta_0 + x_i^T \theta - y_i)^2
\]
We have specify our class of functions and the loss function. Now we need a method to do the optimization. Many modern machine learning methods use SGD to do the optimization.
We say a model generalizes well, if it gets good predictions on test data.
What we really care about is the performance of the model on future test data. We should not use the performance on the training data to evaluate the quality of the model (because there could be overfitting).
Break & Quiz

Q 2.1: When we train a model, we are

- A. Optimizing the parameters and keeping the features fixed.
- B. Optimizing the features and keeping the parameters fixed.
- C. Optimizing the parameters and the features.
- D. Keeping parameters and features fixed and changing the predictions.
Break & Quiz

Q 2.1: When we train a model, we are

- A. Optimizing the parameters and keeping the features fixed.
- B. Optimizing the features and keeping the parameters fixed.
- C. Optimizing the parameters and the features.
- D. Keeping parameters and features fixed and changing the predictions.
Break & Quiz

Q 2.1: When we train a model, we are

• A. Optimizing the parameters and keeping the features fixed.
• B. Optimizing the features and keeping the parameters fixed) (Feature vectors $x_i$ don’t change during training).
• C. Optimizing the parameters and the features. (Same as B)
• D. Keeping parameters and features fixed and changing the predictions. (We can’t train if we don’t change the parameters)
Break & Quiz

- **Q 2.2**: You have trained a classifier, and you find there is significantly higher loss on the test set than the training set. What is likely the case?

  - A. You have accidentally trained your classifier on the test set.
  - B. Your classifier is generalizing well.
  - C. Your classifier is generalizing poorly.
  - D. Your classifier is ready for use.
Break & Quiz

- **Q 2.2**: You have trained a classifier, and you find there is significantly higher loss on the test set than the training set. What is likely the case?

- A. You have accidentally trained your classifier on the test set.
- B. Your classifier is generalizing well.
- **C. Your classifier is generalizing poorly.**
- D. Your classifier is ready for use.
Break & Quiz

- **Q 2.2:** You have trained a classifier, and you find there is significantly higher loss on the test set than the training set. What is likely the case?

- **A.** You have accidentally trained your classifier on the test set. *(No, this would make test loss lower)*
- **B.** Your classifier is generalizing well. *(No, test loss is high means poor generalization)*
- **C.** Your classifier is generalizing poorly.
- **D.** Your classifier is ready for use. *(No, will perform poorly on new data)*
Break & Quiz

• Q 2.3: You have trained a classifier, and you find there is significantly lower loss on the test set than the training set. What is likely the case?

• A. You have accidentally trained your classifier on the test set.
• B. Your classifier is generalizing well.
• C. Your classifier is generalizing poorly.
• D. Your classifier needs further training.
Break & Quiz

- **Q 2.3**: You have trained a classifier, and you find there is significantly lower loss on the test set than the training set. What is likely the case?

- **A. You have accidentally trained your classifier on the test set.** (This is very likely, loss will usually be the lowest on the data set on which a model has been trained)
- **B. Your classifier is generalizing well.**
- **C. Your classifier is generalizing poorly.**
- **D. Your classifier needs further training.**
Linear Regression

Simplest type of regression problem.

- **Inputs:** \((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\)
  - \(x\)'s are vectors, \(y\)'s are scalars.
  - “Linear”: predict a linear combination of \(x\) components + intercept

\[
f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \theta_0 + x^T \theta
\]

- **Want:** optimal parameters
Usually, we wrap intercept by doing:

1. Let $\bar{\theta}$ be the concatenation of $[\theta_0 \ \theta]$
2. Let $\bar{x}$ be the concatenation of $[1 \ x]$

Then $f(x) = \langle \bar{x}, \bar{\theta} \rangle$. For simplicity, we reload the notation (writing $\bar{x}$ as $x$ and $\bar{\theta}$ as $\theta$), and then write $f(x) = \langle x, \theta \rangle$.

$X$: each column is a data point.
Can do SGD.

When $X^T X$ is invertible we can get a closed form solution.
How Good are the Optimal Parameters?

Now we have parameters $\hat{\theta} = (X^T X)^{-1} X^T y$

- How good are they?
- Predictions are $f(x_i) = \hat{\theta}^T x_i = ((X^T X)^{-1} X^T y)^T x_i$
- Errors ("residuals")
  \[
  |y_i - f(x_i)| = |y_i - \hat{\theta}^T x_i| = |y_i - ((X^T X)^{-1} X^T y)^T x_i|
  \]
- If data is linear, residuals are 0. Almost never the case!
Train/Test for Linear Regression?

So far, residuals measure error on train set

- Sometimes that’s all we care about (Fixed Design LR)
  - Data is deterministic.
  - Goal: find best linear relationship on dataset

- Or, create a test set and check (Random Design LR)
  - Common: assume data is $y = \theta^T x + \varepsilon$
  - The more noise, the less linear

0-mean Gaussian noise
For binary classification with labels \{0,1\}, we can squash the linear function output to a probability value in \([0,1]\) and use that to model \(p(y=1|x)\).

**Linear Regression \rightarrow Classification?**

What if we want the same idea, but \(y\) is 0 or 1?
- Need to convert the \(\theta^T x\) to a probability in \([0,1]\)

\[
p(y = 1|x) = \frac{1}{1 + \exp(-\theta^T x)}
\]

**Logistic function**

Why does this work?
- If \(\theta^T x\) is really big, \(\exp(-\theta^T x)\) is really small \(\Rightarrow p\) close to 1
- If really negative exp is huge \(\Rightarrow p\) close to 0

“Logistic Regression”